Fast campus

제주도 도로 교통량 예측 AI 분석 발표

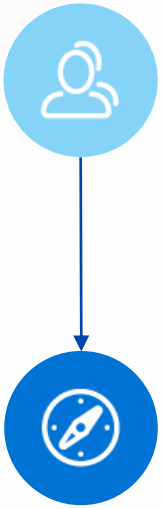
제주도의 교통 정보로부터 도로 교통량 회귀 예측

팀명: 감귤보이즈 2팀(김재승, 우현, 김정현, 배재한)

주최: 제주도 테크노 파크, 제주특별자치도/주관:데이콘(DACON)

<목차>

주제 선정



프로젝트 배경

프로젝트 목표

1. 주제선정: 프로젝트 배경 및 목표

제주도 교통정보데이터를 바탕으로 현재도로의 교통량을 예측 해보기!!

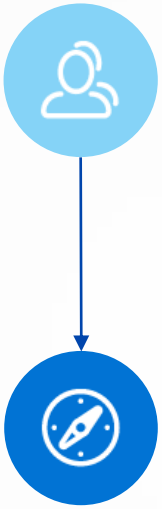
현재 시각 제주도 도로교통 체증 원인 및 중요도

1. 제주도 내 주민등록수는 2022년 기준으로 약 70만명으로 집계 됨.
2. 연평균 1.5% 정도 매년 증가되는 추세~~.
3. 내국인 및 외국인관광객을 고려하면 전체 상주 인구는 90만명을 넘을 것으로 추정.

출처: 한겨레

머신러닝 모델을 이용한 교통량(평균속도) 예측 분석

데이터 설명



프로젝트 배경

프로젝트 목표

2. 데이터 설명(1) 데이터 EDA

내부 데이터(Dataset.Info)

1. train.csv [파일]

- 4,701,217개의 데이터

-base_date(날짜) : 2021.09.01 - 2022.07.31

-day_of_week(요일) : 월,화,수,목,금,토,일

-base_hour(시간대) : 0 - 23

-lane_count(차로수) : 1, 2, 3

-road_rating(도로등급) : 103, 106, 107

-road_name(도로이름) : 61 가지의 도로명(일반국도12호선, -(결측치) 등....)

-multi_linked(중용구간 여부) : 0, 1

-connect_code(연결로 코드) : 0, 103

-maximum_speed_limit(최고속도제한) : 30, 50, 60, 70, 80

-vehicle_restricted(통과제한차량) : 0

-weight_restricted(통과제한하중) :0/ 43200/ 32400/ 50000

-height_restricted(통과제한높이) : 0

-road_type(도로유형) : 0, 3

-start_node_name, end_node_name : 487 도로, 장소명

-start_latitude, start_longitude : 586 가지 출발점 위,경도

-end_latitude, end_longitude : : 586 가지 도착점 위,경도

-start_turn_restricted : 있음, 없음

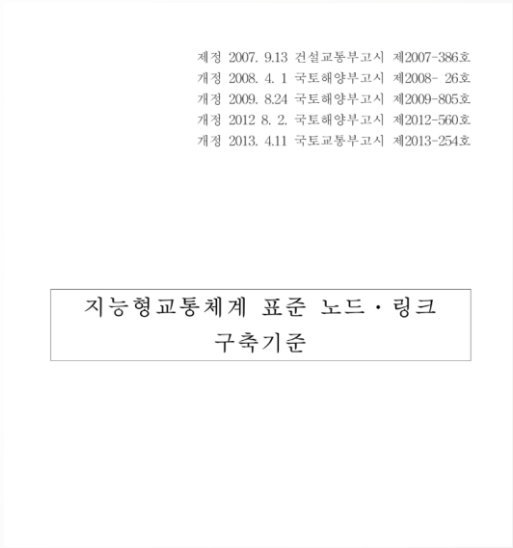
-end_turn_restricted : 있음, 없음

-target : 평균속도(예측해야할 값)

외부 데이터(Dataset.Info)

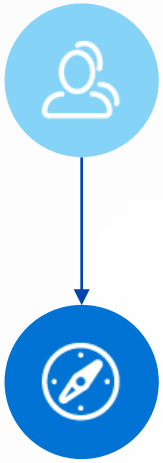
1. Distance: 지리 공간 파생변수 도로의 위치 및 구간거리
2. Airport_distance: 제주공항까지 거리의 제주도 권역별 구분(제주도, 서귀포)
3. Road_mean: GPS 정보를 사용해서 road 구분한 평균
4. Slope: 각 도로의 경사도 계산
5. Tourist: 제주도 관광객 입도 현황
6. Is_camera: 무인 교통단속 카메라 위치
7. 그외....

국토 해양부 데이터 자료


제정 2007. 9.13 건설교통부고시 제2007-386호
개정 2008. 4. 1 국토해양부고시 제2008- 26호
개정 2009. 8.24 국토해양부고시 제2009-805호
개정 2012 8. 2. 국토해양부고시 제2012-560호
개정 2013. 4.11 국토교통부고시 제2013-254호

지능형교통체계 표준 노드·링크
구축기준

데이터 설명



데이터 EDA

- 데이터 분포
- 결측치 처리
- 데이터 시각화

Feature Engineering

- 범주형 인코딩
- 피처 선택

2. 데이터 설명(1) 데이터 EDA-

국토 해양부 데이터 자료 상세 설명

마. 노드 속성정보에 활용하는 코드값은 아래와 같다.

영문명	한글명	코드값	설명
NODE_TYPE	노드유형	101 102 103 104 105 108	도로교차점 도로시·종점 속성변화점 도로시설물 행정경계 IC 및 JC
TURN_P	회전제한유무	0 1	무 유

영문명	한글명	코드값	설명
TURN_TYPE	회전제한유형	001 002 003 011 012 101 102 103	비보호회전 버스만회전 회전금지 U-TURN P-TURN 좌회전금지 직진금지 우회전금지
TURN_OPER	회전제한운영	0 1	전일제 시간제

영문명	한글명	코드	코드정보
ROAD_RANK	도로등급	101 102 103 104 105 106 107 108	고속국도 도시고속국도 일반국도 특별·광역시도 국가지원지방도 지방도 시·군도 기타
ROAD_TYPE	도로유형	000 001 002 003 004	일반도로 고가차도 지하차도 교량 터널
ROAD_USE	도로사용여부	0 1	사용 미사용
MULTI_LINK	중용구간여부	0 1	독립구간 중용구간

자료를 보고 떠오른 생각

1. 위도 경도는 지도 관련 데이터?? -> Folium??

2. 중용구간의 의미는??

. - 일반국도의 도로 구간은 전용구간과 중용구간으로 나눌 수 있다. 전용구간이란하나의 노선이 도로를 전적으로 사용하는 구간을 말하며 2개 이상의 노선이 도로의 일정 구간을 공동으로 사용하는 구간을 중용구간이라한다.

3. 통과제한하중: 과적으로인한 도로 파괴과적...

4. 통과제한차량: 무게가 우선 부피도 제재도 한다.

5. 도로등급: 도로상태를 보고 정함.

6. 차량제한에 따라 도로차선

   - 승용차: 1차선~3차선

   - 화물차는 3차선

7. 일반도로/고속도로/추월선(고속도로에서 1차선에서만 가능)

9. 노선: 교통 기관이 통과하는 출발 지점과 목적 지점을 잇는 선이다.

   예시) 철도 노선, 버스 노선, 항공 노선, 항로 등이 있다.

10. 데이터 조사기간: 2021.09.31 ~ 2022.07.31 거의 1년치인데 관광객을 목표로해서 저 날짜기간을 둔것인가??

2. 데이터 설명(1) 데이터 EDA- 데이터 분포

데이터 분석 세트

```python
# 데이터분석 세트
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter
import matplotlib as mpl
import os
import plotly.express as px
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import OrdinalEncoder
from sklearn.model_selection import GridSearchCV
import gc
from sklearn.preprocessing import LabelEncoder
from sklearn.cluster import KMeans
import math
from sklearn.model_selection import StratifiedKFold as kfold
from sklearn.metrics import mean_absolute_error
from lightgbm import LGBMRegressor
from xgboost import XGBRegressor
from catboost import catRegressor
```

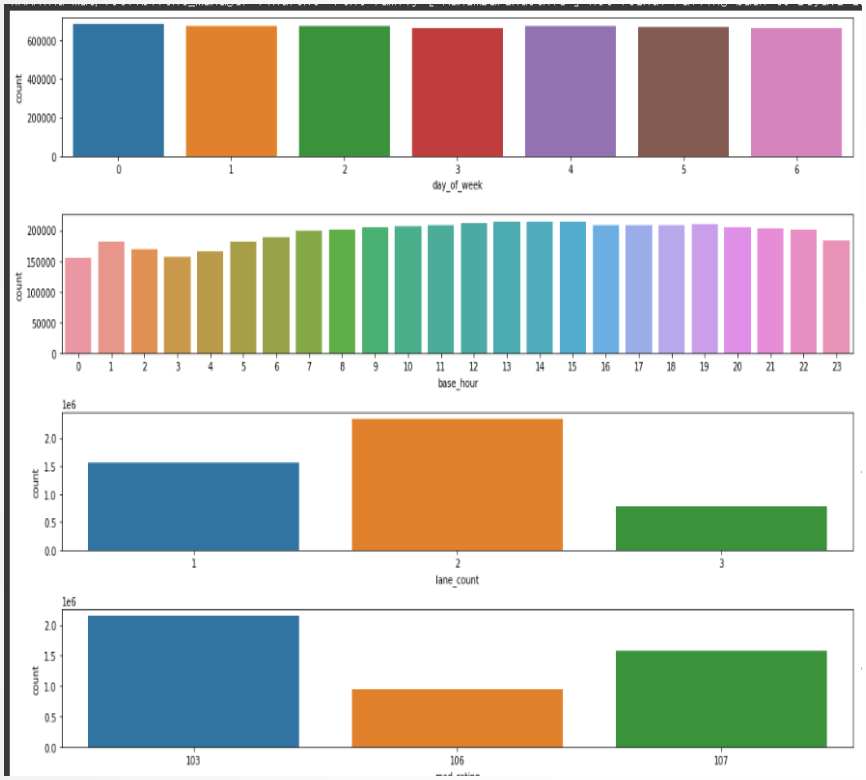
데이터의 개수

```
id = 4701217
base_date = 281
day_of_week = 7
base_hour = 24
lane_count = 3
road_rating = 3
road_name = 61
multi_linked = 2
connect_code = 2
maximum_speed_limit = 6
vehicle_restricted = 1
weight_restricted = 4
height_restricted = 1
road_type = 2
start_node_name = 487
start_latitude = 586
start_longitude = 586
start_turn_restricted = 2
end_node_name = 487
end_latitude = 586
end_longitude = 586
end_turn_restricted = 2
target = 102
```

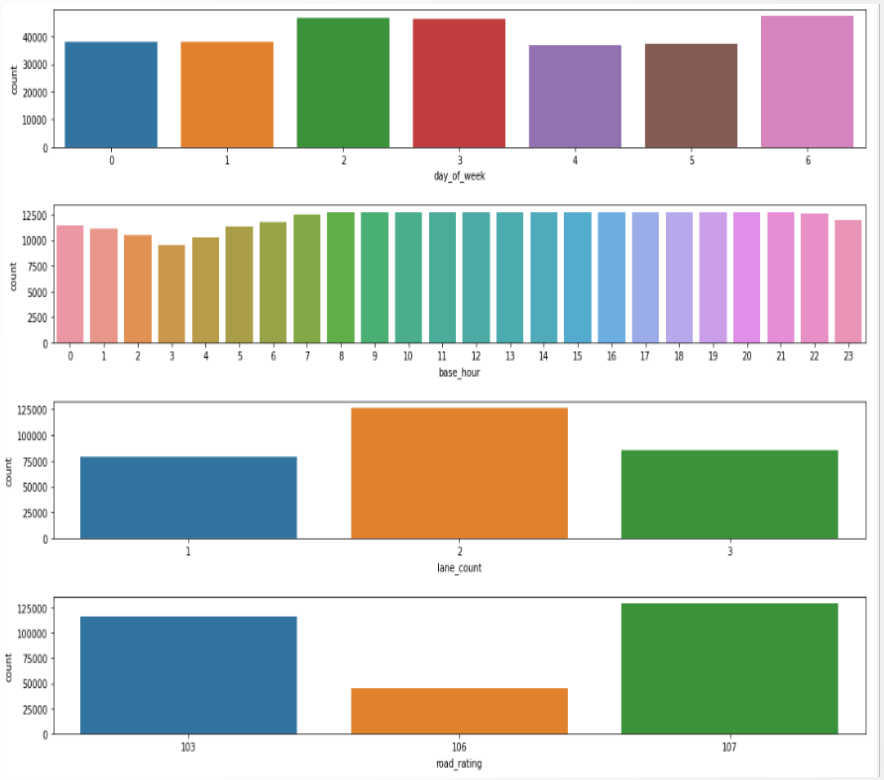
2. 데이터 설명(1) 데이터 EDA- 데이터 분포

Train/Test Set의 데이터 분포 비교

Train data



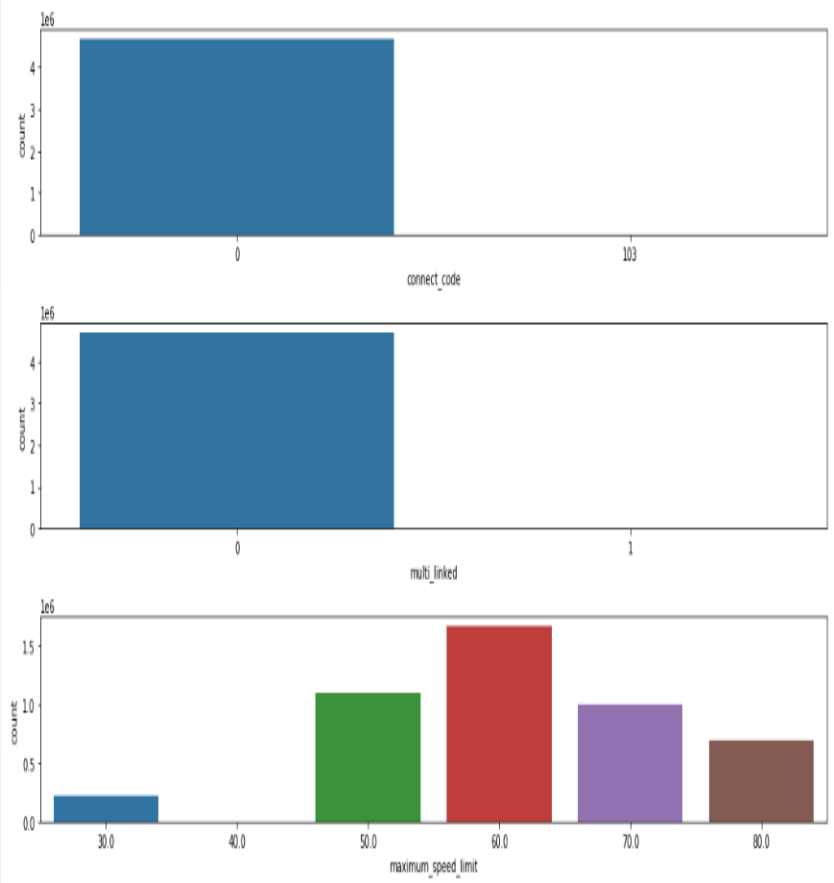
Test data



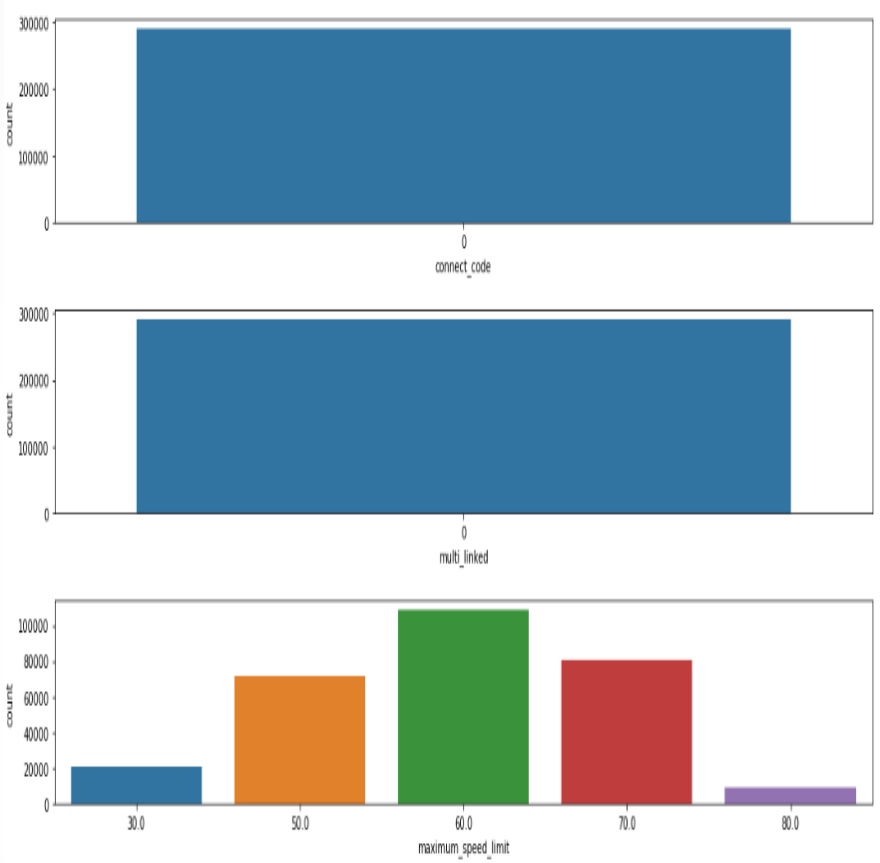
2. 데이터 설명(1) 데이터 EDA- 데이터 분포

Train/Test Set의 데이터 분포 비교

Train data



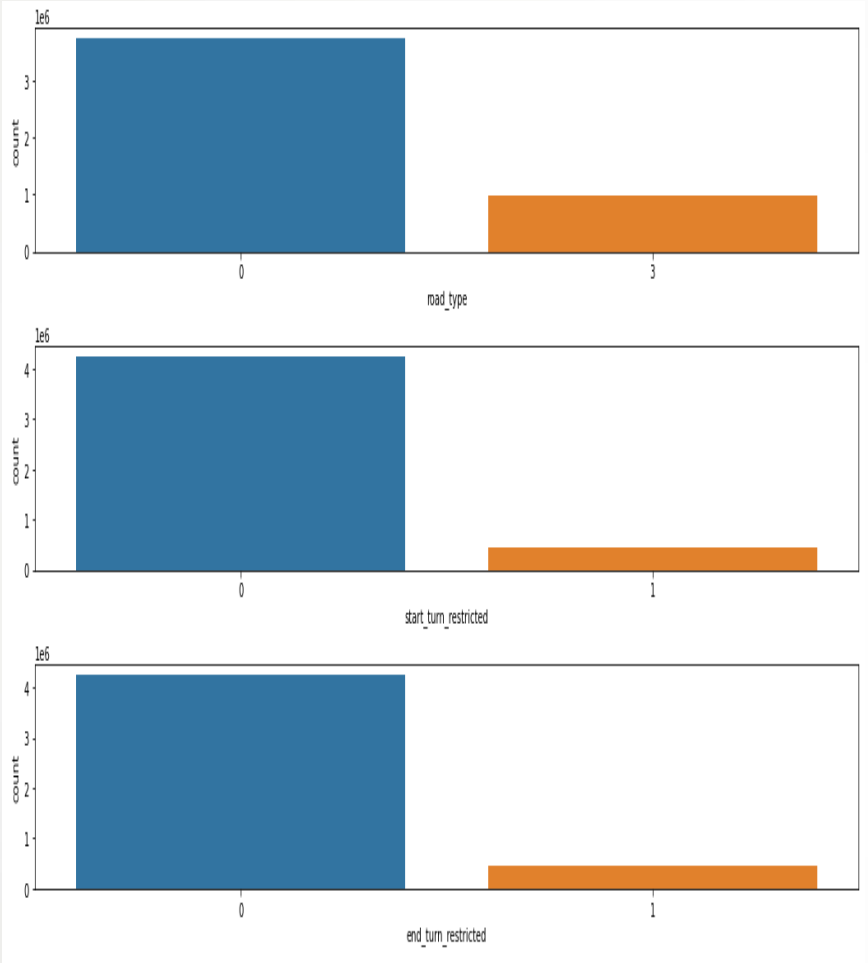
Test data



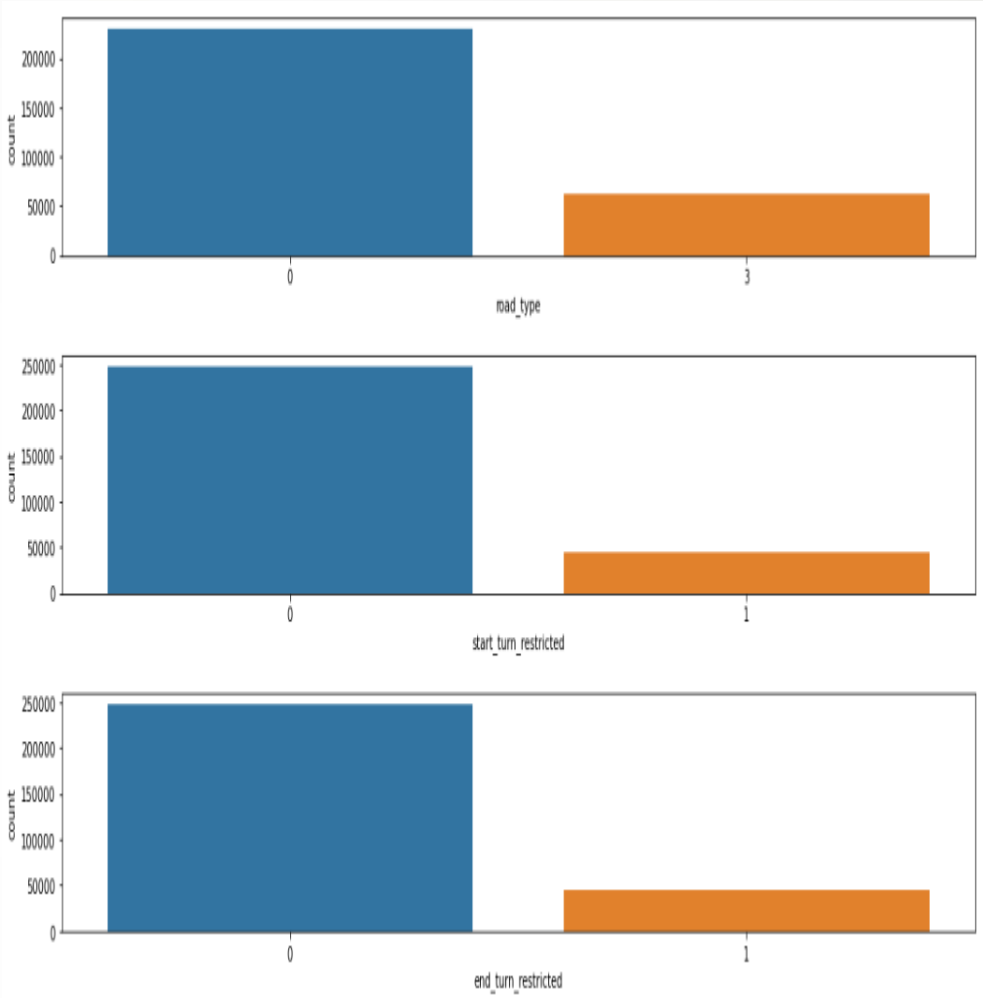
2. 데이터 설명(1) 데이터 EDA- 데이터 분포

Train/Test Set의 데이터 분포 비교

Train data



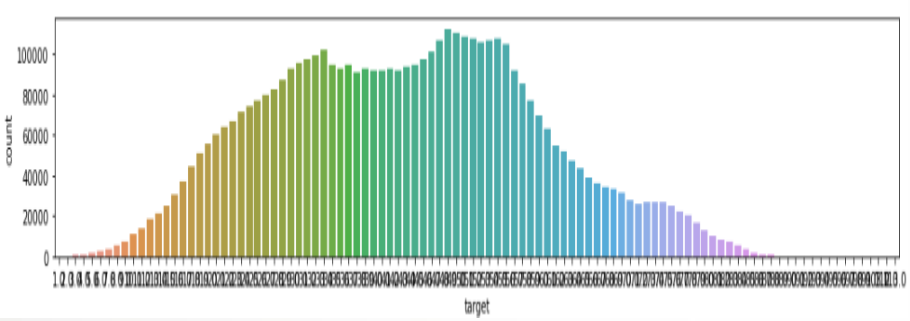
Test data



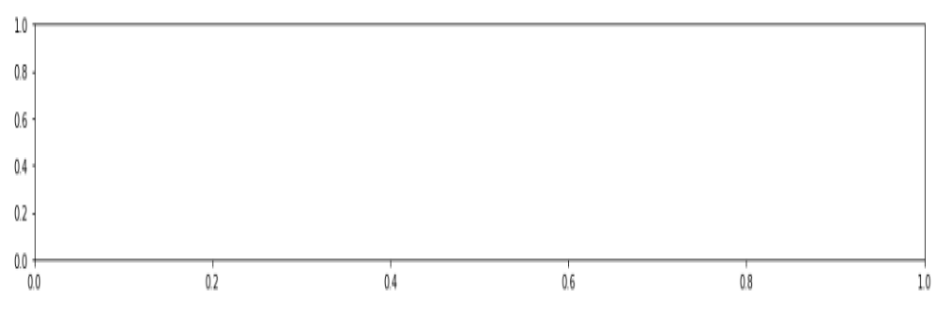
2. 데이터 설명(1) 데이터 EDA- 데이터 분포

Train/Test Set의 데이터 분포 비교

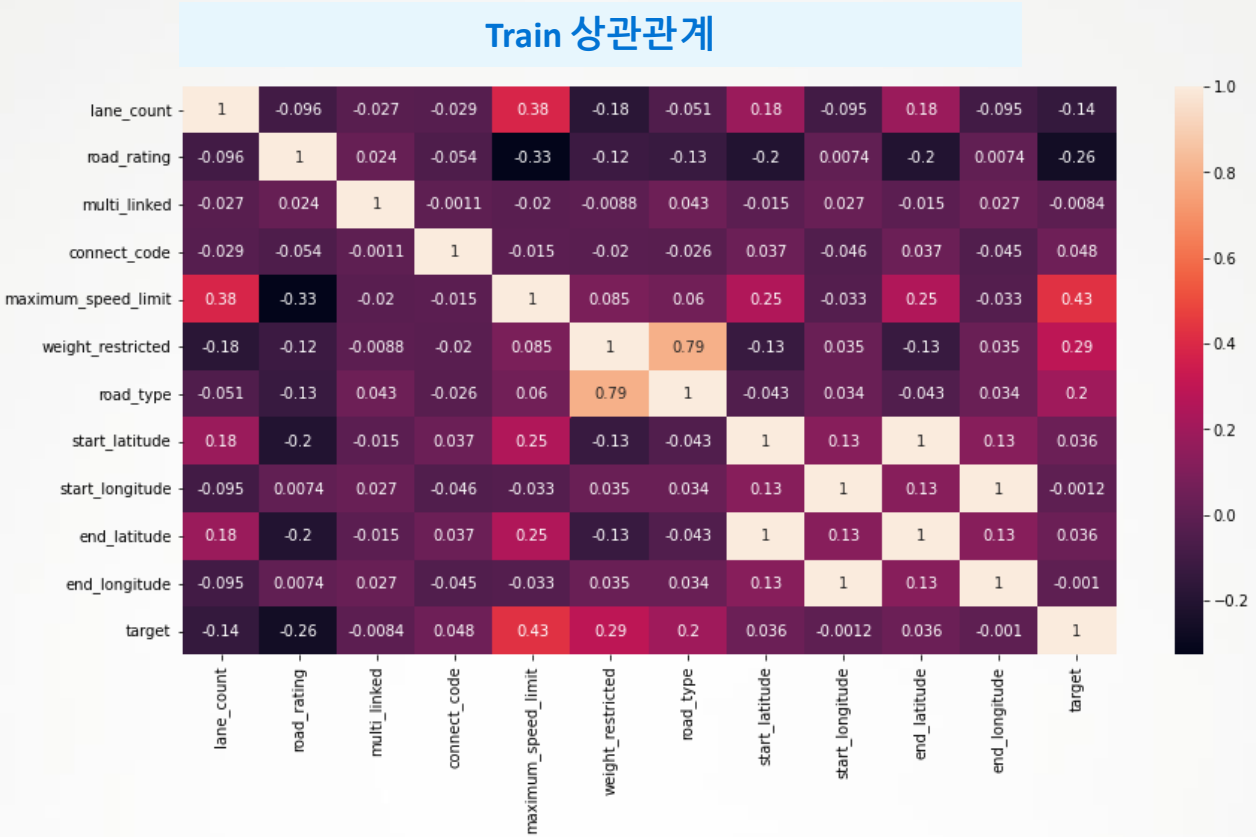
Train data



Test data



2. 데이터 설명(1) 데이터 EDA- 데이터 분포



1. 예측해야하는 target값과의 상관관계가 높은 것중에 가장 눈에 띄는 컬럼은 maximum_speed_limit입니다. 최고속도 제한이 높을 수록 양의 상관관계가 있다는 것을 알 수 있습니다(직관적으로도)

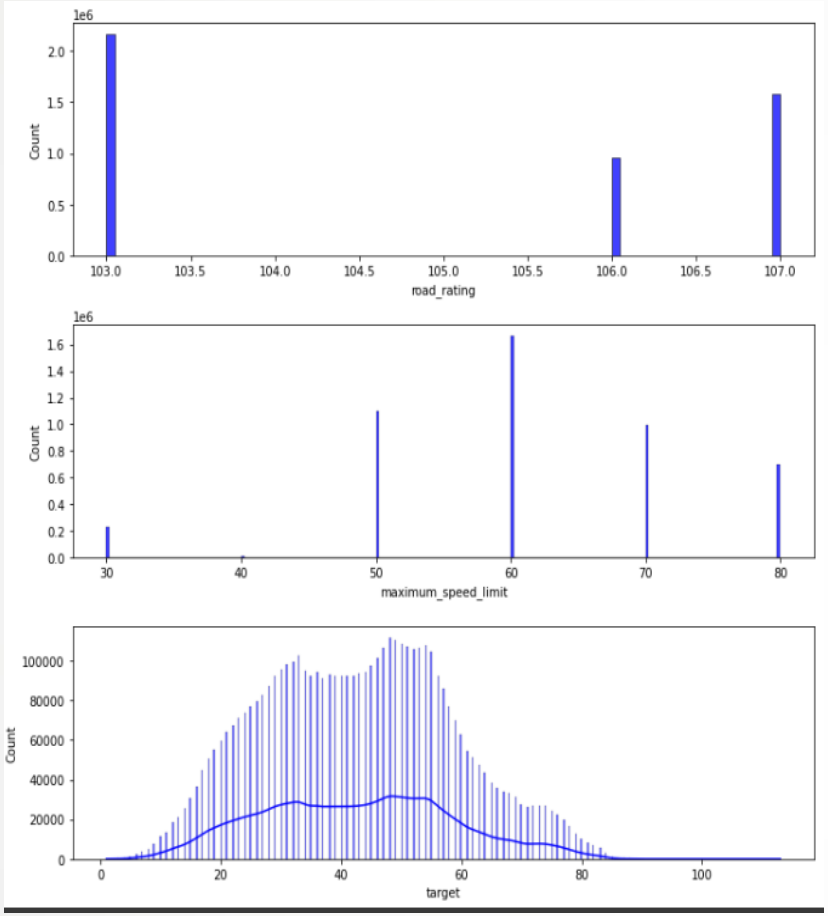
2. lane_count(차로 수)는 많을수록 속도가 더 높을 것 같음에도 maximum_speed_limit와도 관련.

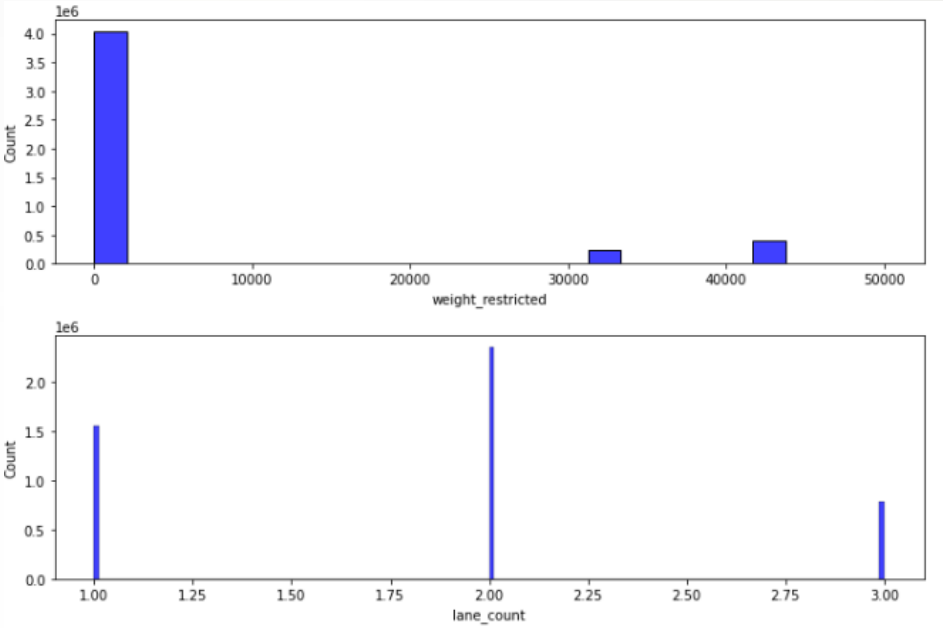
3. weight_restricted(통과 제한 하중)은 target과의 양의 상관관계가 나왔습니다. Road_rating, road_type은 상관관계가 있으나 데이터를 보아야 원인을 유추할 수 있을 것으로 예상됨.

2. 데이터 설명(1) 데이터 EDA- 데이터 분포

Train/Test Set의 데이터 분포 비교

Train data



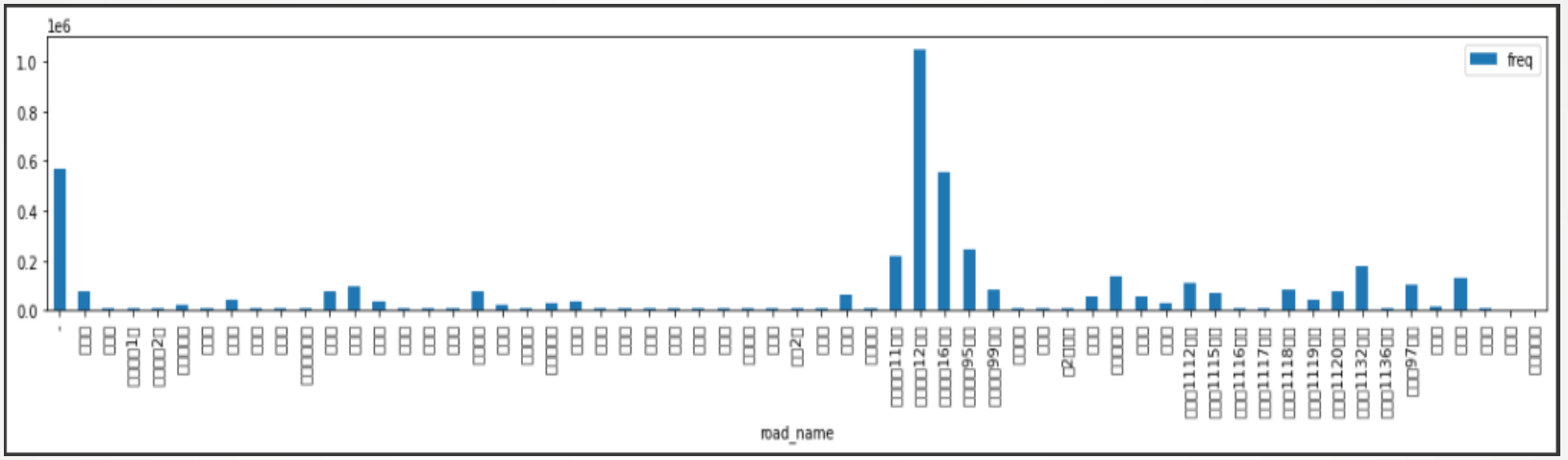


[고려해본변수들]

1. road_rating
2. Weight_restricted
3. Maximum_speed_limit
4. lane_count
5. target

2. 데이터 설명(1) 데이터 EDA- 데이터 분포

Train data

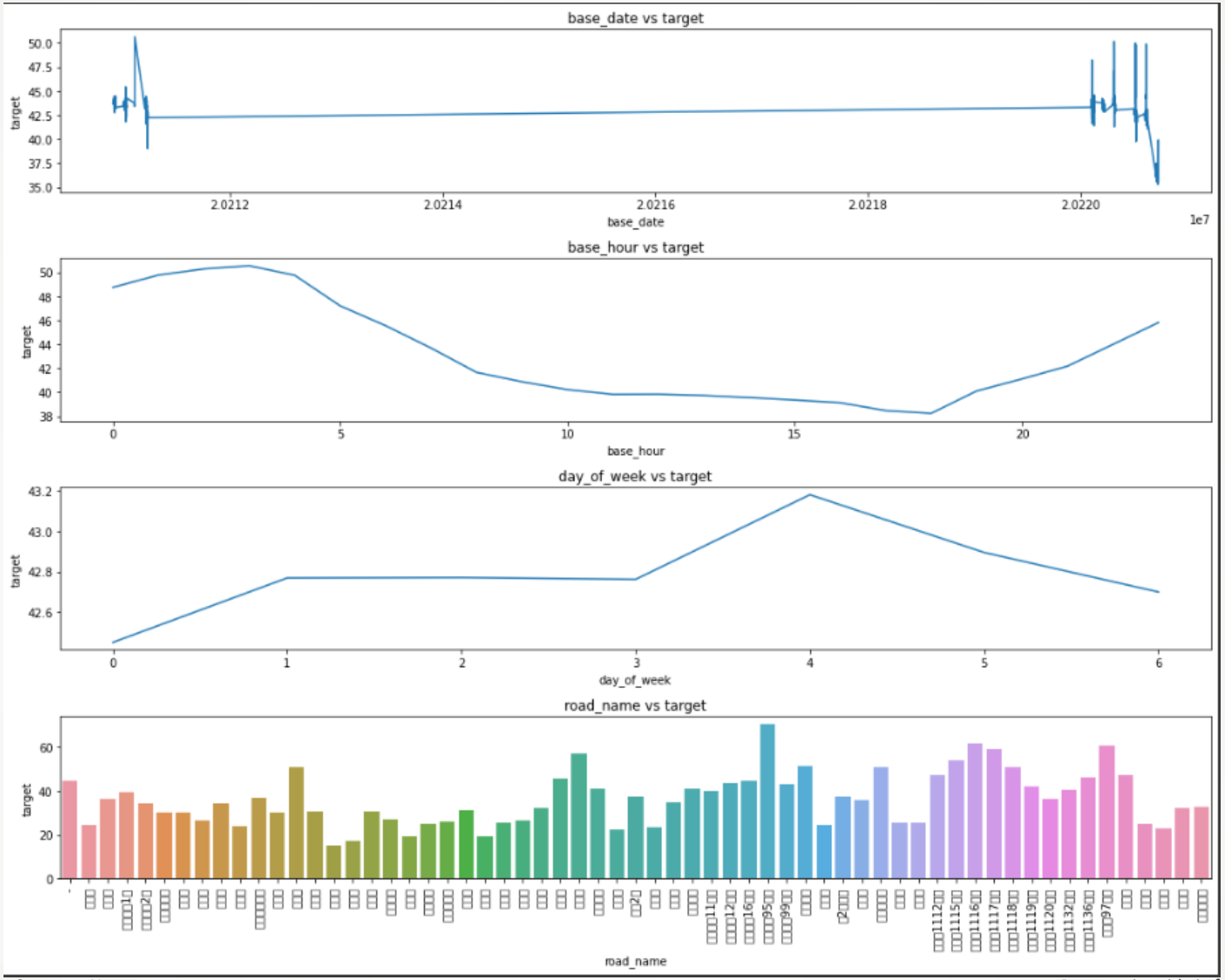


일반국도 11번
일반국도 12번
일반국도 16번
일반국도 95번
일반국도 99번

제일 많이 측정이 됨. => 교통량 원활??

2. 데이터 설명(1) 데이터 EDA- 데이터 분포

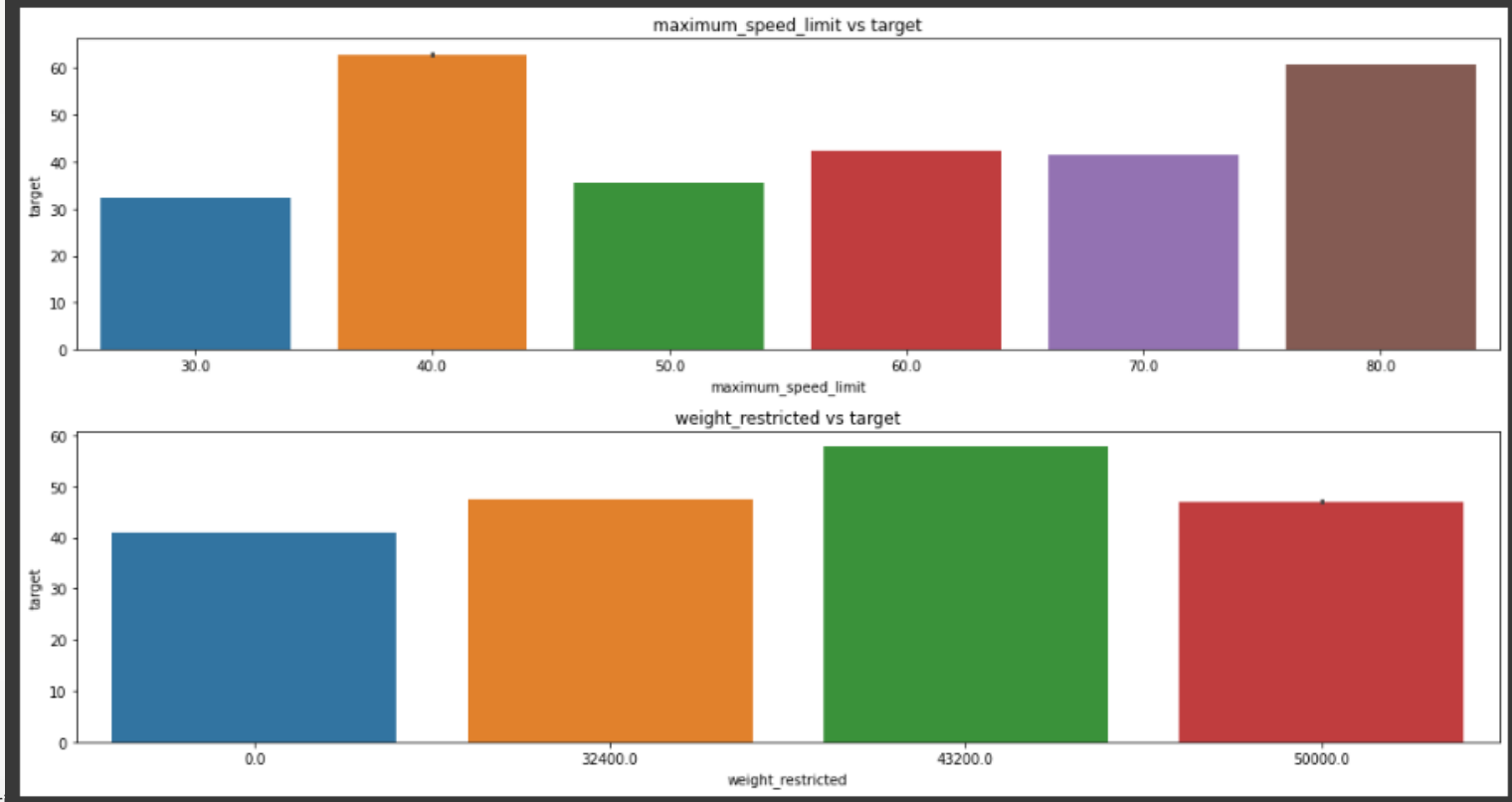
Train 데이터 분포



2. 데이터 설명(1) 데이터 EDA- 데이터 분포

Train 데이터 분포

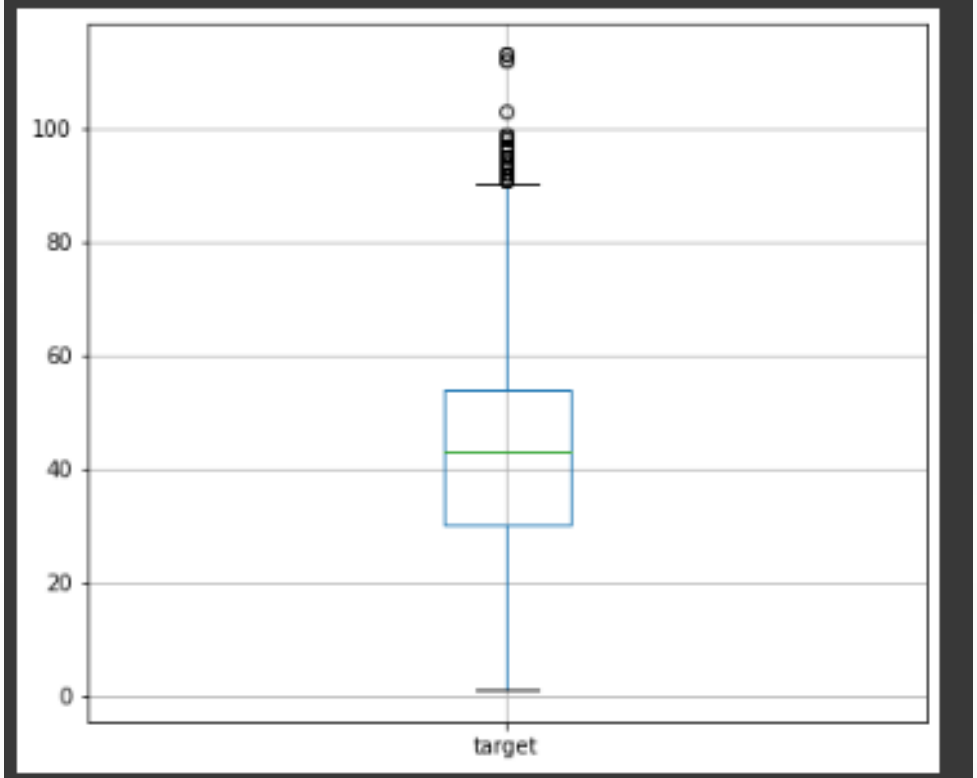
```
# 최대속도제한,무게제한 VS t=target
# 최대속도제한에서 30km에서 평균속도 30km
# 무게제한이 없는 곳일수록 교통혼잡하고
# 최대속도제한인 30,50, 60, 70인 곳에서 교통혼잡이 일어난다는 것을 알았다.
fig, axes = plt.subplots(2, figsize=(15,8))

sns.barplot(x='maximum_speed_limit',y='target',data=train,ax=axes[0]).set(title='maximum_speed_limit vs target')
sns.barplot(x='weight_restricted',y='target',data=train,ax=axes[1]).set(title='weight_restricted vs target')

plt.tight_layout()
plt.show()
```



2. 데이터 설명(1) 데이터 EDA- 데이터 분포

Target 데이터 분포

```
1 plt.figure(figsize=(7,6))
2 boxplot = train.boxplot(column=["target"])
3 plt.show()
```



2. 데이터 설명(1) 데이터 EDA- 데이터 분포

Train 데이터 분포
road_name/최대 속도/최저속도 비교

```
1 # 도로별 max와 min이 서로다른 구간
2 result.loc[result["maximum_speed_limit(max)"] != result["maximum_speed_limit(mi
```

road_name	maximum_speed_limit(max)	maximum_speed_limit(min)
-	80.0	30.0
남조로	60.0	50.0
동홍로	60.0	50.0
산서로	60.0	50.0
새서귀로	60.0	30.0
서사로	50.0	30.0
신대로	70.0	30.0
연북로	50.0	30.0
일반국도11호선	70.0	30.0
일반국도12호선	80.0	30.0
일반국도16호선	80.0	30.0
일반국도95호선	80.0	40.0
일반국도99호선	70.0	30.0
중산간서로	70.0	50.0
중앙로	70.0	60.0
지방도1112호선	60.0	30.0
지방도1118호선	70.0	50.0
지방도1120호선	60.0	30.0
지방도1132호선	70.0	50.0

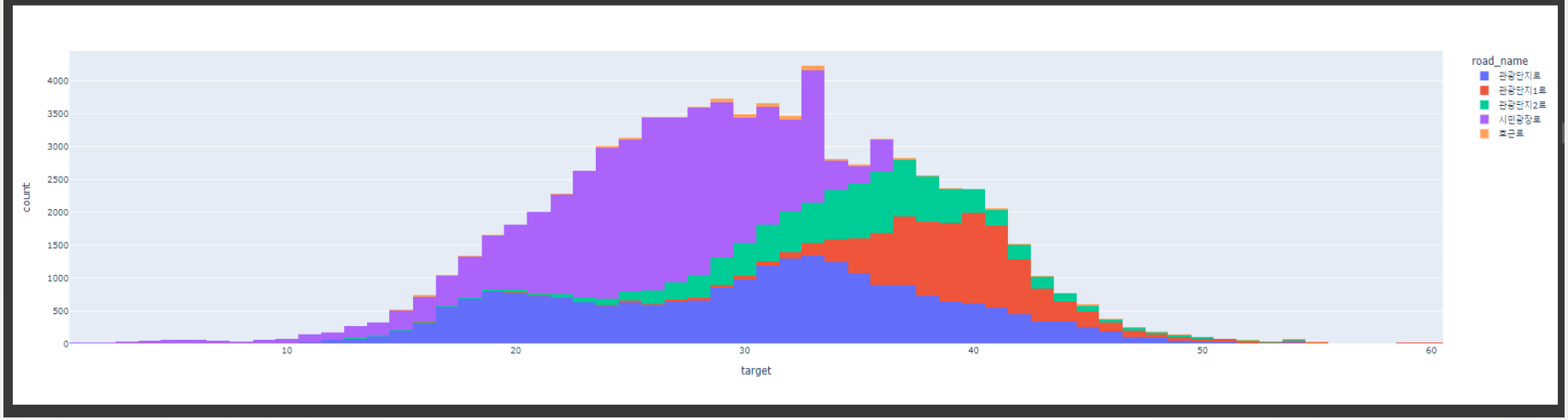
```
1 # maximum speed : 30, 50, 60, 70 일때 확인
2 # 위의 네개 값에서 교통혼잡이 많이 일어날거라 추론함.
3
4 trafic_jam_30 = result.loc[result["maximum_speed_limit(max)"] <= 30]
5 trafic_jam_30
```

road_name	maximum_speed_limit(max)	maximum_speed_limit(min)
관광단지1로	30.0	30.0
관광단지2로	30.0	30.0
관광단지로	30.0	30.0
시민광장로	30.0	30.0
호근로	30.0	30.0

2. 데이터 설명(1) 데이터 EDA- 데이터 분포

Train 데이터 분포

```
1 jam_list_30 = trafic_jam_30.index.tolist()
2 df_temp = train.loc[train["road_name"].isin(jam_list_30)]
3 fig = px.histogram(df_temp, x="target", color="road_name")
4 fig.show()
```

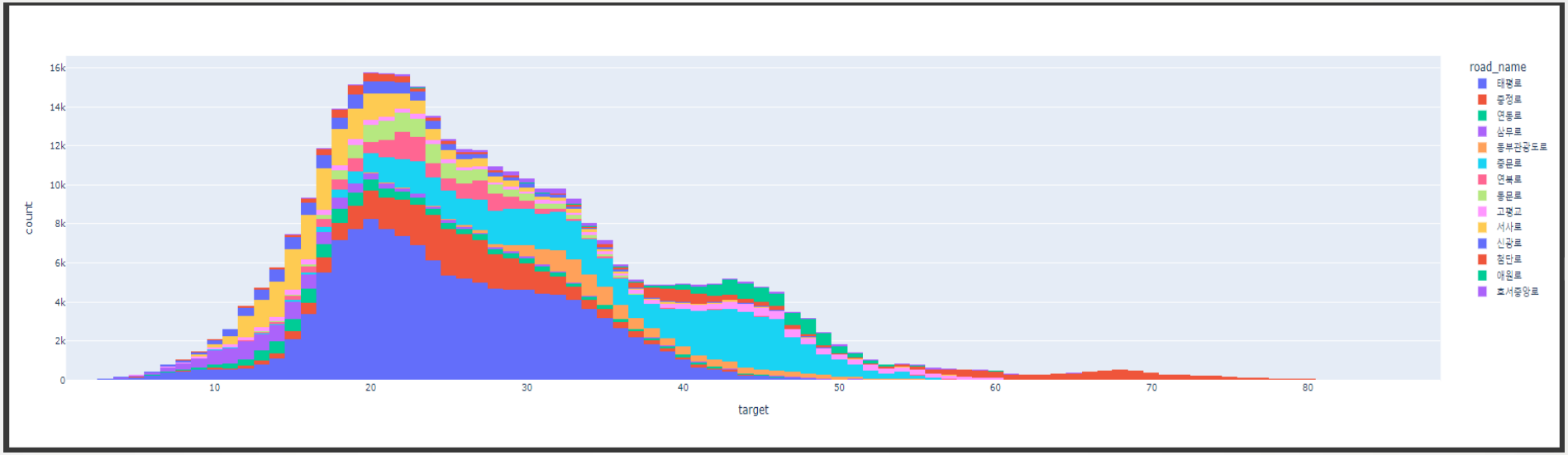


(road_name/최대속도/ 최저속도 30km 비교)

road_name	maximum_speed_limit(max)	maximum_speed_limit(min)
관광단지1로	30.0	30.0
관광단지2로	30.0	30.0
관광단지로	30.0	30.0
시민광장로	30.0	30.0
호근로	30.0	30.0

2. 데이터 설명(1) 데이터 EDA- 데이터 분포

Train 데이터 분포

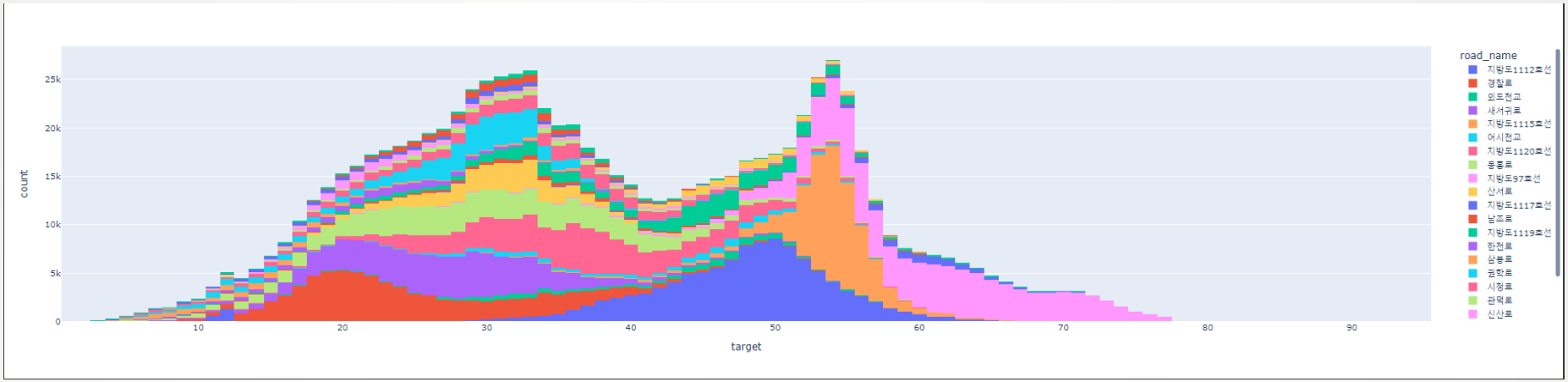


(road_name/최대속도/ 최저속도 50km 비교)

road_name	maximum_speed_limit(max)	maximum_speed_limit(min)
고평교	50.0	50.0
동문로	50.0	50.0
동부관광도로	50.0	50.0
삼무로	50.0	50.0
서사로	50.0	30.0
신광로	50.0	50.0
애원로	50.0	50.0
연동로	50.0	50.0
연북로	50.0	30.0
중문로	50.0	50.0
중정로	50.0	50.0
첨단로	50.0	50.0
태평로	50.0	50.0
호서중앙로	50.0	50.0

2. 데이터 설명(1) 데이터 EDA- 데이터 분포

Train 데이터 분포

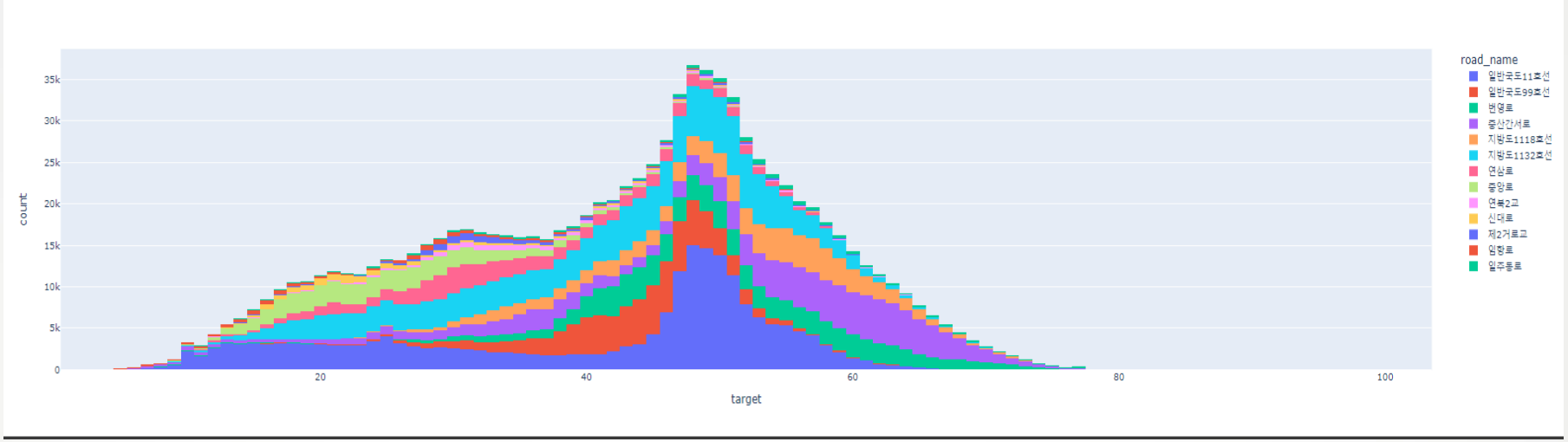


(road_name/최대속도/ 최저속도 60km 비교)

road_name	maximum_speed_limit(max)	maximum_speed_limit(min)
경찰로	60.0	60.0
관덕로	60.0	60.0
권학로	60.0	60.0
남조로	60.0	50.0
동홍로	60.0	50.0
산서로	60.0	50.0
삼봉로	60.0	60.0
삼성로	60.0	60.0
새서귀로	60.0	30.0
수영장길	60.0	60.0
시청로	60.0	60.0
신산로	60.0	60.0
아봉로	60.0	60.0
어시천교	60.0	60.0
외도천교	60.0	60.0
지방도1112호선	60.0	30.0
지방도1115호선	60.0	60.0
지방도1117호선	60.0	60.0
지방도1119호선	60.0	60.0
지방도1120호선	60.0	30.0
지방도1136호선	60.0	60.0
지방도97호선	60.0	60.0
한천로	60.0	60.0

2. 데이터 설명(1) 데이터 EDA- 데이터 분포

Train 데이터 분포
(road_name/최대속도/ 최저속도 비교)

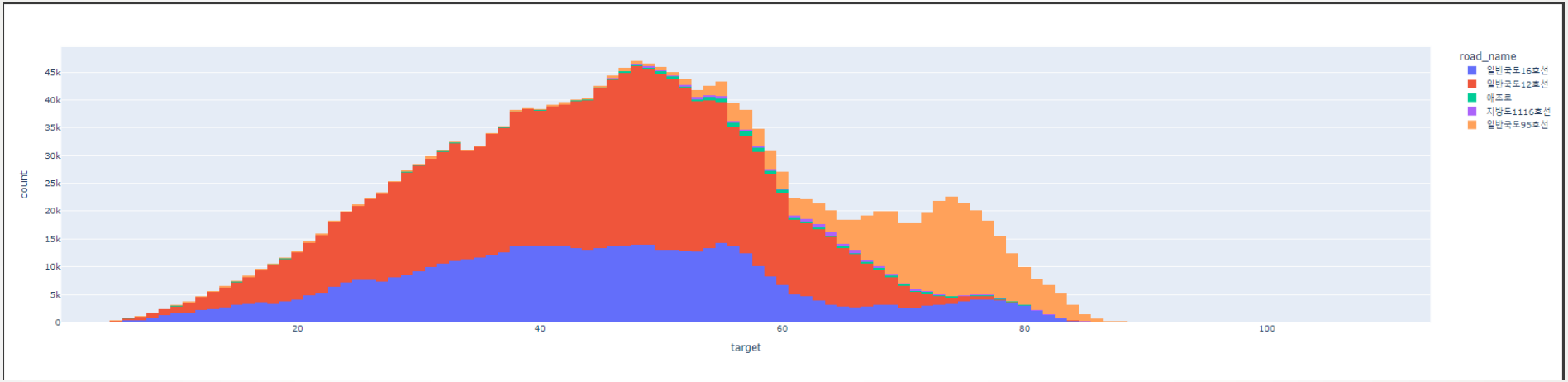


(road_name/최대속도/ 최저속도 70km비교

road_name	maximum_speed_limit(max)	maximum_speed_limit(min)
번영로	70.0	70.0
신대로	70.0	30.0
연북2교	70.0	70.0
연삼로	70.0	70.0
일반국도11호선	70.0	30.0
일반국도99호선	70.0	30.0
일주동로	70.0	70.0
임항로	70.0	70.0
제2거로교	70.0	70.0
중산간서로	70.0	50.0
중앙로	70.0	60.0
지방도1118호선	70.0	50.0
지방도1132호선	70.0	50.0

2. 데이터 설명(1) 데이터 EDA- 데이터 분포

Train 데이터 분포

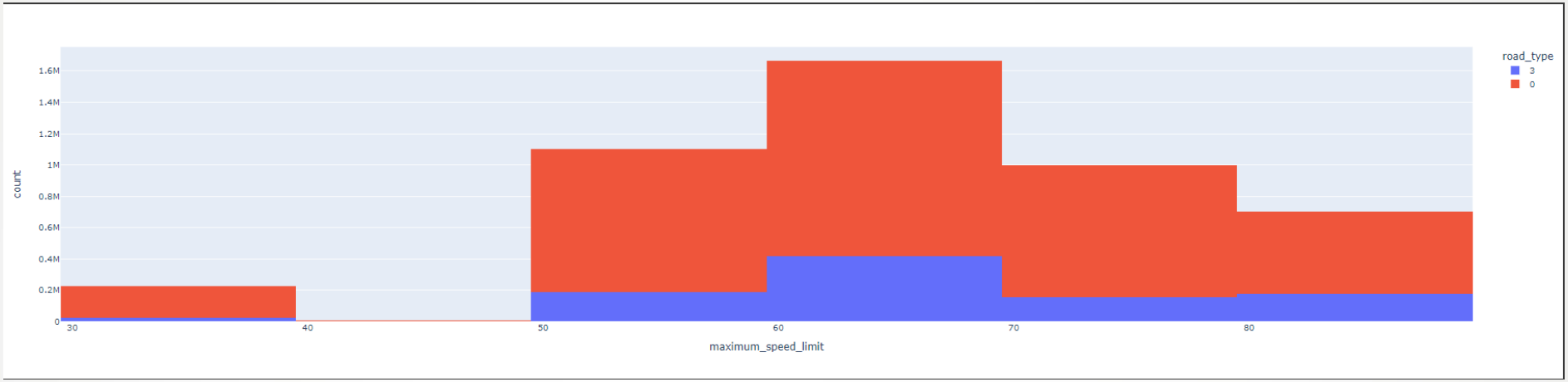


(road_name/최대속도/ 최저속도 80km 비교)

road_name	maximum_speed_limit(max)	maximum_speed_limit(min)
-	80.0	30.0
애조로	80.0	80.0
일반국도12호선	80.0	30.0
일반국도16호선	80.0	30.0
일반국도95호선	80.0	40.0
지방도1116호선	80.0	80.0

2. 데이터 설명(1) 데이터 EDA- 데이터 분포

Train 데이터 분포
(road_type/maximum_speed_limit)



2. 데이터 설명(1) 데이터 EDA- 데이터 분포

요일별 target 통계

day_of_week	count
금	684024
수	675583
목	674070
일	673632
토	669767
화	662498
월	661643

요일별 target 통계

day_of_week	base_hour	count
금	13	32115
일	19	31734
금	15	31715
목	14	31706
토	11	31697
목	16	31666
화	10	31646
토	12	31544
금	8	31492
토	13	31485

요일별/ 새벽시간대

day_of_week	base_hour	count
목	4	23714
일	2	23603
금	3	23342
목	3	23325
일	23	23208
	4	23183
금	4	23173
수	3	22871
금	0	22843
화	3	22430
월	3	22392
	0	22269
토	4	22098
	2	21967
화	0	21633
일	3	21488
수	0	21465
일	0	21351
토	0	20739
	3	20090

2. 데이터 설명(1) 데이터 EDA- 데이터 분포

Train 데이터 상관관계

```
# 컬럼 간 상관관계(pearson)
corr_mat = train.corr(method="pearson").abs()
sorted_mat = corr_mat.unstack().sort_values(ascending=False)
sorted_mat = sorted_mat[sorted_mat.lt(1)]

sorted_mat.head(50)
# road_type와 weight_restricted 상관관계 높다.
# maximum_speed_limit과 road_rating, lane_count, target 어느정도 상관(0.3~0.4)
# 결론: 교통혼잡의 원인 = 도로시설의 미비이다.
```

Train 데이터 상관관계

1. road_type와 weight_restricted 상관관계 높다.
2. maximum_speed_limit과 road_rating, lane_count, target 어느정도 상관(0.3~0.4)

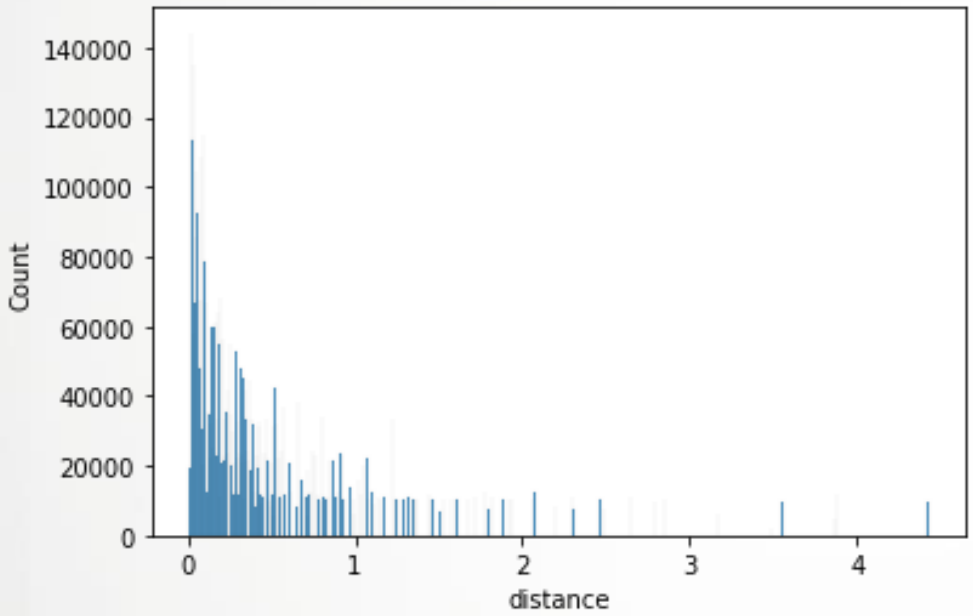
3. 결론: 교통혼잡의 원인 = 도로시설의 미비이다.

```
end_latitude         start_latitude         0.999180
start_latitude       end_latitude           0.999180
start_longitude      end_longitude          0.999143
end_longitude        start_longitude        0.999143
weight_restricted    road_type              0.770532
road_type            weight_restricted      0.770532
target               maximum_speed_limit    0.421441
maximum_speed_limit  target                 0.421441
lane_count           maximum_speed_limit    0.377472
maximum_speed_limit  lane_count             0.377472
                     road_rating            0.342178
road_rating          maximum_speed_limit    0.342178
target               road_rating            0.324382
road_rating          target                 0.324382
target               weight_restricted      0.292849
weight_restricted    target                 0.292849
maximum_speed_limit  start_latitude         0.257554
start_latitude       maximum_speed_limit    0.257554
end_latitude         maximum_speed_limit    0.257445
maximum_speed_limit  end_latitude           0.257445
end_latitude         road_rating            0.220278
road_rating          end_latitude           0.220278
                     start_latitude         0.220221
start_latitude       road_rating            0.220221
road_type            target                 0.194981
target               road_type              0.194981
start_latitude       lane_count             0.185216
lane_count           start_latitude         0.185216
end_latitude         lane_count             0.184845
lane_count           end_latitude           0.184845
                     weight_restricted      0.181511
weight_restricted    lane_count             0.181511
end_latitude         end_longitude          0.178611
end_longitude        end_latitude           0.178611
start_longitude      start_latitude         0.178577
start_latitude       start_longitude        0.178577
                     end_longitude          0.178507
end_longitude        start_latitude         0.178507
end_latitude         start_longitude        0.178361
start_longitude      end_latitude           0.178361
base_hour            target                 0.159883
target               base_hour              0.159883
                     lane_count             0.149444
lane_count           target                 0.149444
road_rating          weight_restricted      0.148552
weight_restricted    road_rating            0.148552
road_type            road_rating            0.122723
road_rating          road_type              0.122723
weight_restricted    end_latitude           0.110721
end_latitude         weight_restricted      0.110721
dtype: float64
```

2. 데이터 설명(1) 데이터 EDA- 외부데이터 또는 파생변수추가

위도/ 경도차이

```
[ ]  1 # 위도, 경도 차이
     2 train['lat_change'] = train['start_latitude'] - train['end_latitude']
     3 train['lon_change'] = train['start_longitude'] - train['end_longitude']
     4
     5 test['lat_change'] = test['start_latitude'] - test['end_latitude']
     6 test['lon_change'] = test['start_longitude'] - test['end_longitude']
```




```python
# 두지점 사이의 거리
from math import radians, cos, sin, asin, sqrt

def haversine(row):
    """
    Calculate the great circle distance between two points
    on the earth (specified in decimal degrees)
    """
    # convert decimal degrees to radians
    lon1 = row['start_longitude']
    lat1 = row['start_latitude']
    lon2 = row['end_longitude']
    lat2 = row['end_latitude']

    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])
    # haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    km = 6367 * c
    return km
```

2. 데이터 설명(1) 데이터 EDA- 외부데이터 또는 파생변수추가

제주공항까지 거리 파생변수

```python
# 제주공항까지 거리
def haversine_airport(row):
    """
    Calculate the great circle distance between two points
    on the earth (specified in decimal degrees)
    """
    # convert decimal degrees to radians
    lon1 = 126.4913534
    lat1 = 33.5104135
    lon2 = (row['start_longitude'] + row['end_longitude']) / 2
    lat2 = (row['start_latitude'] + row['end_latitude']) / 2

    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])
    # haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    km = 6367 * c
    return km

train['airport_distance'] = train.apply(haversine_airport, axis=1)
test['airport_distance'] = test.apply(haversine_airport, axis=1)
```

2. 데이터 설명(1) 데이터 EDA- 외부데이터 또는 파생변수추가

제주 권역별 구분하여 파생 변수 추가

제주도 권역별 구분하여 변수 추가

- 제주시 도심 : 126.4531517 ~ 126.5900257 , 33.4670429 ~
- 서귀포 도심 : 126.3972753 ~ 126.6076604 , ~ 33.2686052

```
[ ]  1 # 출발지점 권역
     2 mask_jj_start = (train['start_longitude'] > 126.4531517) & (train['start_longitude']< 126.5900257) & (train['start_latitude'] > 33.4670429)
     3 mask_jj_end = (train['end_longitude'] > 126.4531517) & (train['end_longitude']< 126.5900257) & (train['end_latitude'] > 33.4670429)
     4
     5 mask_sgp_start = (train['start_longitude'] > 126.3972753) & (train['start_longitude']< 126.6076604) & (train['start_latitude'] < 33.2686052)
     6 mask_sgp_end = (train['end_longitude'] > 126.3972753) & (train['end_longitude']< 126.6076604) & (train['end_latitude'] < 33.2686052)
```

```
 1 train['center_start'] = 0
 2 test['center_start'] = 0
 3
 4 train.loc[mask_jj_start, 'center_start'] = 1
 5 train.loc[mask_sgp_start, 'center_start'] = 2
 6
 7 test.loc[mask_jj_start, 'center_start'] = 1
 8 test.loc[mask_sgp_start, 'center_start'] = 2
 9
10 train['center_end'] = 0
11 test['center_end'] = 0
12
13 train.loc[mask_jj_end, 'center_end'] = 1
14 train.loc[mask_sgp_end, 'center_end'] = 2
15
16 test.loc[mask_jj_end, 'center_end'] = 1
17 test.loc[mask_sgp_end, 'center_end'] = 2
```

2. 데이터 설명(1) 데이터 EDA- 외부데이터 또는 파생변수추가

GPS 정보 사용해서 파생 변수 추가

```
[ ]  1 # GPS 정보를 사용해서 road 구분
     2 train['road_code'] = train['start_latitude'].astype(str)+'_'+train['start_longitude'].astype(str)+'_'+train['end_latitude'].astype(str)+'_'+train['end_longitude'].astype(str)
     3 train['road_code'].value_counts()
```

```
33.3058672207151_126.599081327413_33.3082357708673_126.598689775097    6477
33.3082357708673_126.598689775097_33.3058672207151_126.599081327413    6397
33.5014774884938_126.569223187609_33.4968633703578_126.58123009621     6077
33.5016270326083_126.568923085567_33.5014774884938_126.569223187609    6077
33.496710616894_126.581529061335_33.4918481088766_126.591872255149     6075
                                                                       ...
33.2566709359707_126.52441046863_33.2541529264473_126.524330998601      744
33.26127013848_126.524428741607_33.2574097173209_126.52441034435        744
33.2574097173209_126.52441034435_33.2566709359707_126.52441046863       744
33.2574097173209_126.52441034435_33.26127013848_126.524428741607        587
33.2574006381515_126.52574476307_33.2574097173209_126.52441034435       587
Name: road_code, Length: 904, dtype: int64
```

```
[ ]  1 test['road_code'] = test['start_latitude'].astype(str)+'_'+test['start_longitude'].astype(str)+'_'+test['end_latitude'].astype(str)+'_'+test['end_longitude'].astype(str)
     2 test['road_code'].value_counts()
```

```
33.508463678702_126.558231105407_33.5087115227295_126.558702856002    740
33.4937925855376_126.492189386746_33.4923347723675_126.490247073997   740
33.4666066165642_126.454021511351_33.4664333666973_126.454583167413   740
33.4923347723675_126.490247073997_33.4937925855376_126.492189386746   740
33.4658632729266_126.456384480352_33.4664333666973_126.454583167413   740
                                                                      ...
33.3452396554215_126.850113181832_33.3446283972409_126.849278713014     7
33.4857069297096_126.604162168012_33.4886994919865_126.597620980703     7
33.4379464931581_126.73250865826_33.4383285187565_126.732031757687      7
33.4359411786532_126.736248543312_33.4379464931581_126.73250865826      7
33.4288406442461_126.750881044473_33.4359411786532_126.736248543312     7
Name: road_code, Length: 441, dtype: int64
```

2. 데이터 설명(2) Feature engnieering- 변수처리

레이블 인코딩

```
# 범주형 변수 인코딩
# label encoding
str_col = ['day_of_week', 'road_rating', 'road_type', ]
for i in str_col:
    le = LabelEncoder()
    le=le.fit(train[i])
    train[i]=le.transform(train[i])

    for label in np.unique(test[i]):
        if label not in le.classes_:
            le.classes_ = np.append(le.classes_, label)
    test[i]=le.transform(test[i])
```

One hot 인코딩

```
# onehot_encoding
train = pd.get_dummies(train, columns = str_col[1:], drop_first=False)
test = pd.get_dummies(test, columns = str_col[1:], drop_first=False)

train.shape, test.shape
```

```
((4701217, 35), (291241, 34))
```

2. 데이터 설명(2) Feature engineering- 변수처리

Feature Scaling

1. Standard method

```
1 # Feature Scaling
2 from sklearn.preprocessing import StandardScaler
3 scaler = StandardScaler()
4 |
5 scaler.fit(X_train)
6 X_train.loc[:, :] = scaler.transform(X_train)
7 X_test.loc[:, :] = scaler.transform(X_test)
8
9 X_test.head(2)
```

	day_of_week	base_hour	lane_count	maximum_speed_limit	end_latitude	end_longitude	lat_change	lon_change	distance	airport_distance	road_mean
0	-0.991029	0.754498	1.691525	0.72087	1.150794	0.141615	-0.314152	-0.409113	-0.291169	-1.433274	-0.637912
1	1.506502	0.010682	0.237511	0.72087	-1.247087	-0.677032	0.091003	1.808738	0.990841	0.753408	0.387736

1. Min/max scaling.

피처 선택

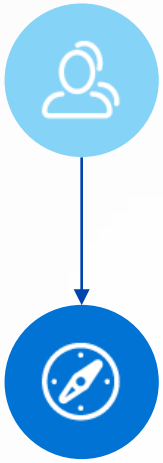
Train

'id'
'base_date'
'target'
'start_node_name'
'end_node_name',
'multi_linked'
'connect_code',
'start_latitude'
'start_longitude',
'center_start'
'center_end',
'weight_restricted', 'road_rating_0',
'road_rating_1',
'road_rating_2',
'road_type_0',
'road_type_1',
'start_turn_restricted',
'end_turn_restricted',
'road_min', 'road_max',
'road_std'
'road_name'
'road_code'

```
# 변수 선택
y_train = train['target']

X_train = train.drop(['id','base_date', 'target', 'start_node_name', 'end_node_name', 'multi_linked', 'connect_code',
                      'start_latitude', 'start_longitude',
                       'center_start', 'center_end',
                      'weight_restricted', 'road_rating_0',
                      'road_rating_1', 'road_rating_2', 'road_type_0', 'road_type_1',
                      'start_turn_restricted','end_turn_restricted',
                      'road_min', 'road_max', 'road_std',
                      'road_name', 'road_code'], axis=1)

X_test = test.drop(['id','base_date', 'start_node_name', 'end_node_name', 'multi_linked', 'connect_code',
                    'start_latitude', 'start_longitude',
                     'center_start', 'center_end',
                    'weight_restricted', 'road_rating_0',
                    'road_rating_1', 'road_rating_2', 'road_type_0', 'road_type_1',
                    'start_turn_restricted','end_turn_restricted',
                    'road_min', 'road_max', 'road_std',
                    'road_name', 'road_code'], axis=1)

print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
```

```
(4701217, 11)
(4701217,)
(291241, 11)
```

데이터 분석



모델링 설명

모델 결과

교통량 회귀 예측: 모델링(부스팅 앙상블)

교통 및 도로 구간 변수와 시간 변수를 사용하여 도로의 차량 평균속도 (Km)를 예측하기 위해 설명이 가능한 Tree 기반 boosting 알고리즘 활용

- 성능과 학습 능력이 우수한 Xgboost, LightGBM, Catboost 모델을 각각 학습하고, 평균 앙상블을 적용하여 최종 예측을 시도
- 도로의 차량 평균속도 예측값에 영향을 미치는 각 변수 중요도 영향력에 대한 고찰

교통 및 도로구간 변수 (내부 + 외부 데이터 +파생 변수 추가

+

시간, 날짜 변수

교통 및 도로구간 변수 (내부 + 외부 데이터 +파생 변수 추가

XGBoost

+

LightGBM

+

Catboost

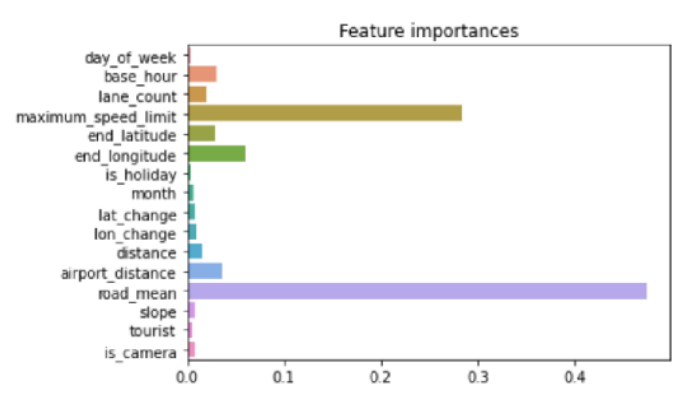
MAE 산출

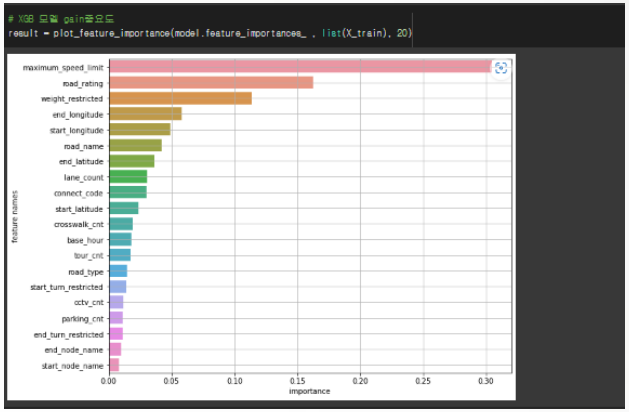
```
RandomForestRegressor mae : 2.9758

ExtraTreeRegressor mae : 3.0209

RandomForestRegressor mae : 2.9757

ExtraTreeRegressor mae : 3.0207

RandomForestRegressor mae : 2.9741

ExtraTreeRegressor mae : 3.0189

mean mae 2.9752
mean mae 3.0202
```

3. 데이터 분석(3) 모델링 결과

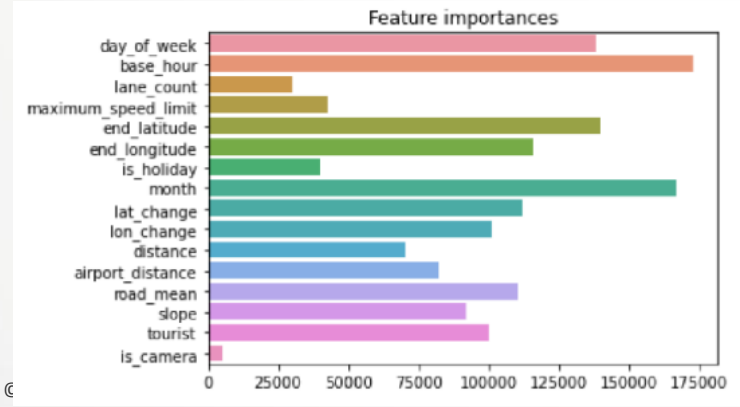
모델링 변수 중요도

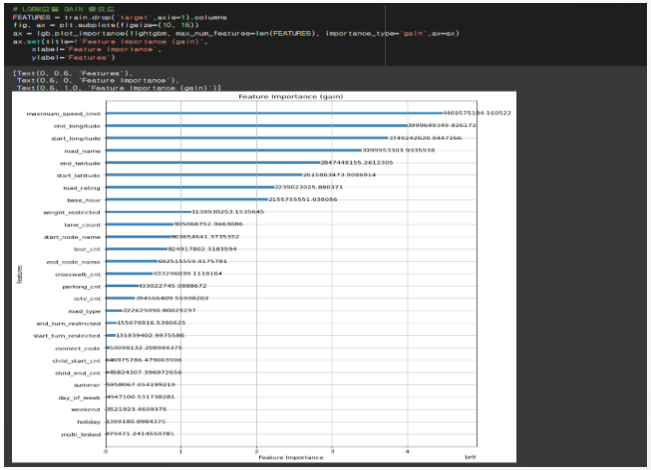
XG Boost Feature importance





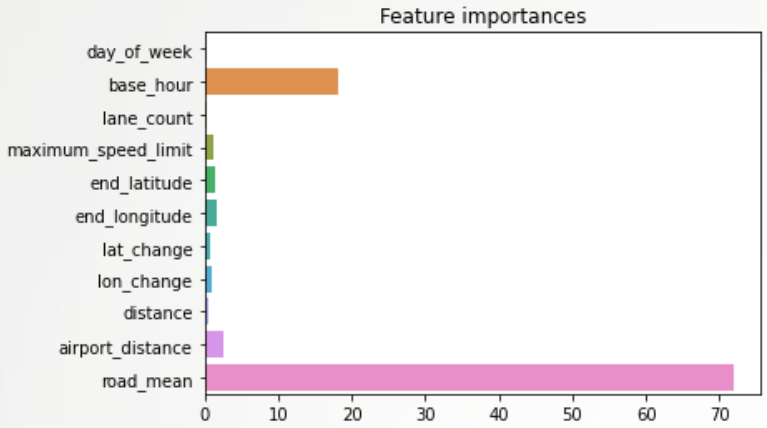
Light GBM Feature importance





3. 데이터 분석(3) 모델링 결과

CatBoost Feature importance




```
bestTest = 3.682752966
bestIteration = 3999

훈련 셋: 3.6823877255151953
검증 셋: 3.682753965950827 7
```

제출 방식

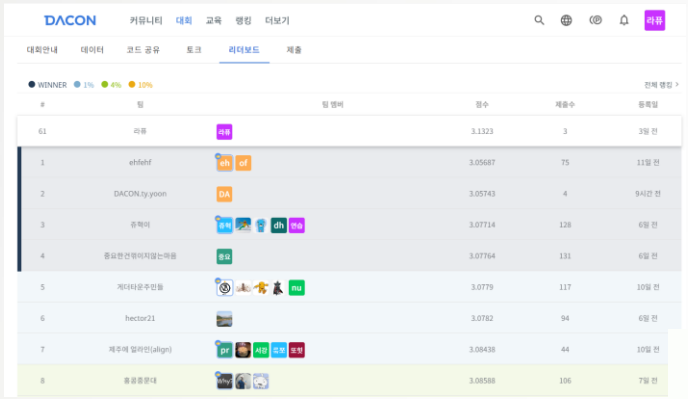
	id	target
0	TEST_000000	26.337253
1	TEST_000001	42.535062
2	TEST_000002	66.478340
3	TEST_000003	38.601332
4	TEST_000004	42.990793
...	...	...
291236	TEST_291236	47.338939
291237	TEST_291237	51.346832
291238	TEST_291238	22.173537
291239	TEST_291239	23.169312
291240	TEST_291240	47.959714

291241 rows × 2 columns

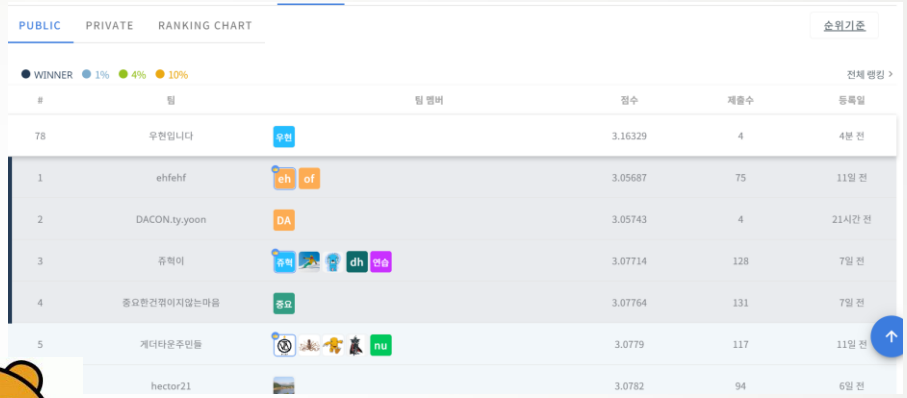
대회 결과

4. 대회 결과

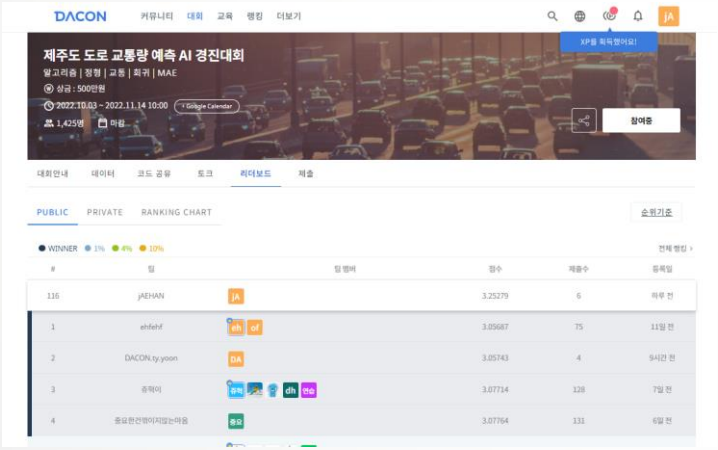
김정현-61등/1420



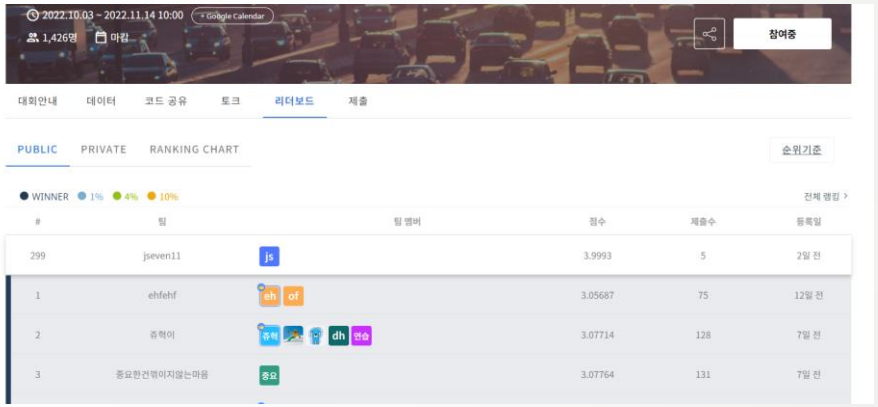
우현-78등/1420



배재한-116등/1420



김재승-229등/1420




Thank you for attention.


Question
Time